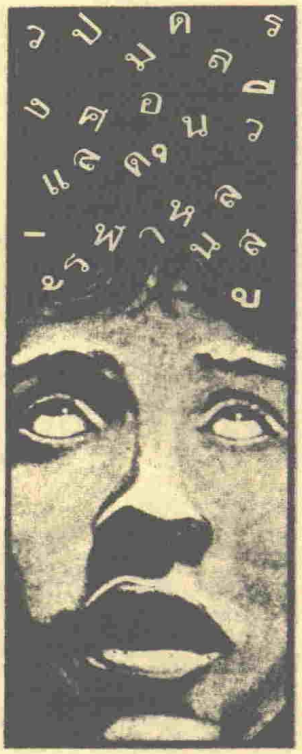
การพัฒนาระบบภาษาไทยระดับแก่น (TKS)



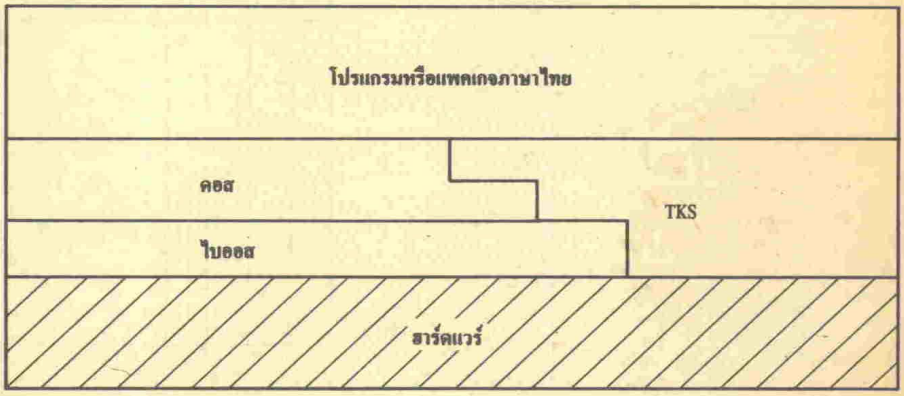
จเร เลิศสุดวิชัย

ในบทความนี้จะเป็นการแนะนำระบบภาษาไทยระดับแก่น (Thai Kernel System, TKS) ซึ่งออกแบบและพัฒนาโดยทีมงานวิจัยห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์ มหาวิทยาลัยเกษตรศาสตร์ พร้อมตัวอย่างการนำเอามาประยุกต์เป็นเครื่องมือใช้งานในการเขียนโปรแกรมแสดงผลภาษาไทย โดยไม่ขึ้นกับไทยการ์ดหรือฮาร์ดแวร์ใด ๆ พร้อมบทความนี้จะมีไลบรารี่ TKS ให้สำหรับผู้อ่านที่สนใจจะนำไปใช้งานรวมทั้งโปรแกรมของบทความนี้ ซึ่งนำฟังก์ชันในไลบารี่ของ TKS มาใช้งานเขียนเป็นฟังก์ชันระบบหน้าต่าง(windows) และระบบเมนู(menu bar และ pop-up menu) บน TKS

ระบบภาษาไทยระดับแก่น (TKS)

ก่อนอื่นเราจะพิจารณาหลักการพื้นฐานของ TKS ที่ถูกสร้างขึ้นเสียก่อน TKS จะควบคุมติดกับฮาร์ดแวร์โดยตรงในเรื่องของภาษาไทย โดยที่ตัวซอฟต์แวร์หรือโปรแกรมที่ใช้ TKS ไม่ต้องยุ่งเกี่ยวเลย ในส่วนของฮาร์ดแวร์ที่สำคัญและเกี่ยวข้องกับภาษาไทยมากที่สุดคือ จอภาพ คีย์บอร์ดและเครื่องพิมพ์ จอภาพนั้นเป็นส่วนแสดงภาษาไทยที่สำคัญที่สุด ระบบ TKS ที่พัฒนาขึ้นมาจะต้องสามารถใช้กับจอภาพชนิดต่าง ๆ ได้ และแสดงผลภาษาไทยได้ถูกต้องตามหลักภาษา ในส่วนคีย์บอร์ด ที่สำคัญคือการอ่านข้อความที่เป็นภาษาไทย ผลลัพธ์ที่ได้จากการอ่านคีย์และผลที่แสดงบนจอภาพจะแตกต่างกัน ปัญหาสำคัญในเรื่องภาษาไทย คือจำนวนคอลัมน์กับจำนวนตัวอักษรอาจจะไม่ตรงกัน ปัญหานี้เราจะพบกันมากในโปรแกรมหรือแพคเกจที่ถูกดัดแปลงให้สามารถแสดงผลภาษาไทยได้ ตัวอย่างเช่น การดัดแปลงโลตัส 123 ที่มีใช้กันอยู่นั้น สามารถแสดงผลภาษาไทยบนจอภาพได้อย่างสวยงามและถูกต้อง แต่เมื่อพิมพ์ผลลัพธ์ออกทางเครื่องพิมพ์ ผลที่ได้นั้นคือจัดเรียงคอลัมน์จะผิดพลาดไปหมด ดังนั้นเครื่องพิมพ์จะเป็นอีกส่วนหนึ่งที่ TKS จะต้องเข้าควบคุม ถ้าเรามองให้ง่ายเพื่อเข้าใจในระบบ TKS จะได้ดังรูปที่ 1

ระบบ TKS ในส่วนของจอภาพเกือบทั้งหมดจะควบคุมฮาร์ดแวร์โดยตรง ส่วนทางด้านคีย์บอร์ดและเครื่องพิมพ์จะมีบางส่วนที่จะผ่านทางไบออสหรือดอสด้วย ระบบ TKS ที่ถูกพัฒนาขึ้นขณะนี้ในส่วนของจอภาพและคีย์บอร์ดได้พัฒนาจนสามารถนำมาใช้งานได้แล้ว ยังขาดทางด้านของเครื่องพิมพ์ ซึ่ง TKS จะต้องสามารถพิมพ์ผลภาษาไทยทางเครื่องพิมพ์ให้ได้ทุกชนิด ไม่ว่าจะเป็นเครื่องพิมพ์ดอตแมตริกซ์ หรือเลเซอร์พรินเตอร์ รหัสภาษาที่ใช้ใน TKS จะใช้รหัสของสมอ.



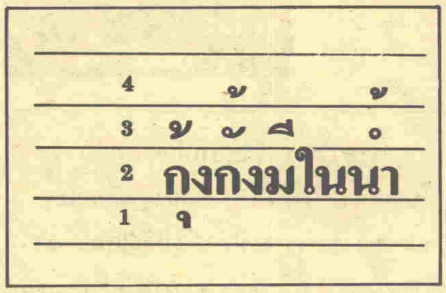
รูปที่ 1 ระบบภาษาไทยระดับแก่น (TKS)

จอภาพ

การแสดงผลภาษาไทยบนจอภาพนั้น TKS จะแสดงผลภาพในโหมดกราฟิก ในขณะนี้ TKS ที่สร้างขึ้นมานั้นสามารถจะแสดงผลบนจอภาพเฮอร์คิวลิส และบนจอ CGA ส่วนจอ EGA, VGA และ MCGA ยังอยู่ในระหว่างพัฒนา ไม่ว่าจอในระบบใด ลักษณะการเรียกใช้ฟังก์ชันจะเหมือนกันทุกประการ TKS สามารถตรวจสอบด้วยตัวเองได้ว่าขณะนี้นั้นใช้จอภาพชนิดใดอยู่ และการแสดงผลก็จะเปลี่ยนไปตามชนิดของจอภาพ ปัญหาที่เกิดขึ้นในภายหลังที่เราพัฒนา TKS ให้แสดงผลบนจอใด ๆ ได้หมดแล้ว ตัวไลบรารี่ของ TKS จะมีขนาดใหญ่มาก เมื่อ Link รวมเข้าโปรแกรมที่เรียกใช้ TKS เอง อาจจะทำให้ได้ขนาดโปรแกรมที่ใหญ่โตมาก ซึ่งอาจจะทำให้หน่วยจำที่เหลือไม่เพียงพอกับการใช้งานโปรแกรมก็ได้ การแก้ไขปัญหาคือทำให้มีการ Install โปรแกรมเพื่อเลือกจอภาพที่ใช้งาน ซึ่งจะสามารถตัดเอาไลบรารี่ของ TKS เฉพาะส่วนของจอภาพชนิดนั้น ๆ ไปใช้งานเท่านั้นเพื่อลดขนาดของโปรแกรม

TKS ที่ทีมงานของห้องปฏิบัติการและวิจัยไมโครฯ สร้างขึ้นนั้น เขียนขึ้นด้วยภาษาแอสเซมบลีและภาษาซี ผสมกันโดยภาษาซีที่ใช้เป็นของไมโครซอฟต์ซี โมเดลที่ใช้เป็นแบบ large ซึ่งผู้ที่จะนำไปใช้งานโปรแกรมนั้นในขั้นนี้จะต้องเป็นโปรแกรมที่เขียนด้วยไมโครซอฟต์ซีเท่านั้น และใช้ large โมเดล

คราวนี้เรามาดูการจัดการจอภาพของ TKS กันก่อนอื่น อย่างที่กล่าวมาแล้วข้างต้นว่า TKS จะแสดงผลภาษาไทยในโหมดกราฟิก ซึ่งใน TKS จะแบ่งการมองจอภาพออกแบบ 2 ระดับ คือ จอภาพฟิสิคัล(Graphic plane) และจอภาพลอจิคัล (Text plane) จอภาพฟิสิคัลคือจอภาพจริง ๆ

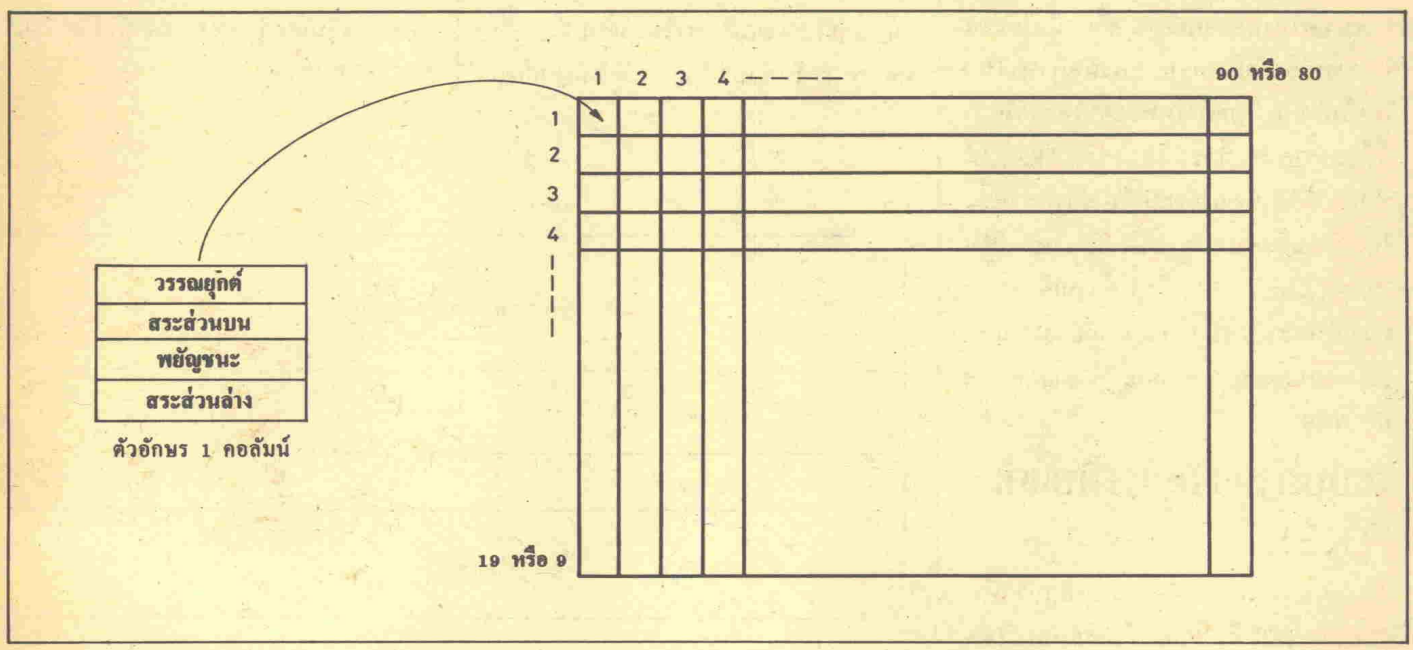


รูปที่ 2 การแสดงผลภาษาไทย 4 ระดับที่ใช้ใน TKS

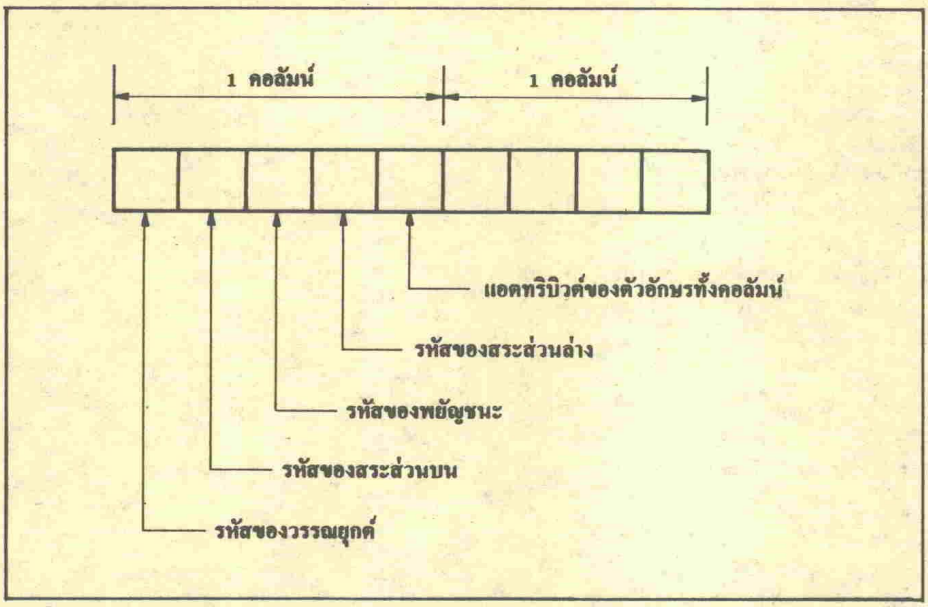
หรือส่วนที่มองเห็นได้ ส่วนจอภาพลอจิคัลนั้นตัว TKS เท่านั้นที่มองเห็นหรือเป็นส่วนที่ TKS ใช้งานสำหรับควบคุมจอภาพฟิสิคัลอีกทีหนึ่ง

จอภาพฟิสิคัล ตัว TKS จะแสดงบนจอภาพกราฟิก โดยการแสดงผลภาษาไทยแบบ 4 ระดับ ดังรูปที่ 2 ขนาดของจอภาพคือจำนวนคอลัมน์และจำนวนบรรทัดของตัวอักษรที่จะแสดงผลนั้นขึ้นอยู่กับชนิดของจอภาพซึ่งให้ความละเอียดไม่เท่ากัน ในจอภาพแบบเฮอร์คิลลิสนั้นจะมีขนาด 90×19 (90 คอลัมน์และ 19 บรรทัด) แต่ในจอภาพแบบ CGA ซึ่งในโหมดกราฟิกจะมีความละเอียดต่ำกว่าจะมีขนาดเพียง 80×9 (80 คอลัมน์ และ 9 บรรทัด) เท่านั้น ดังในรูปที่ 3

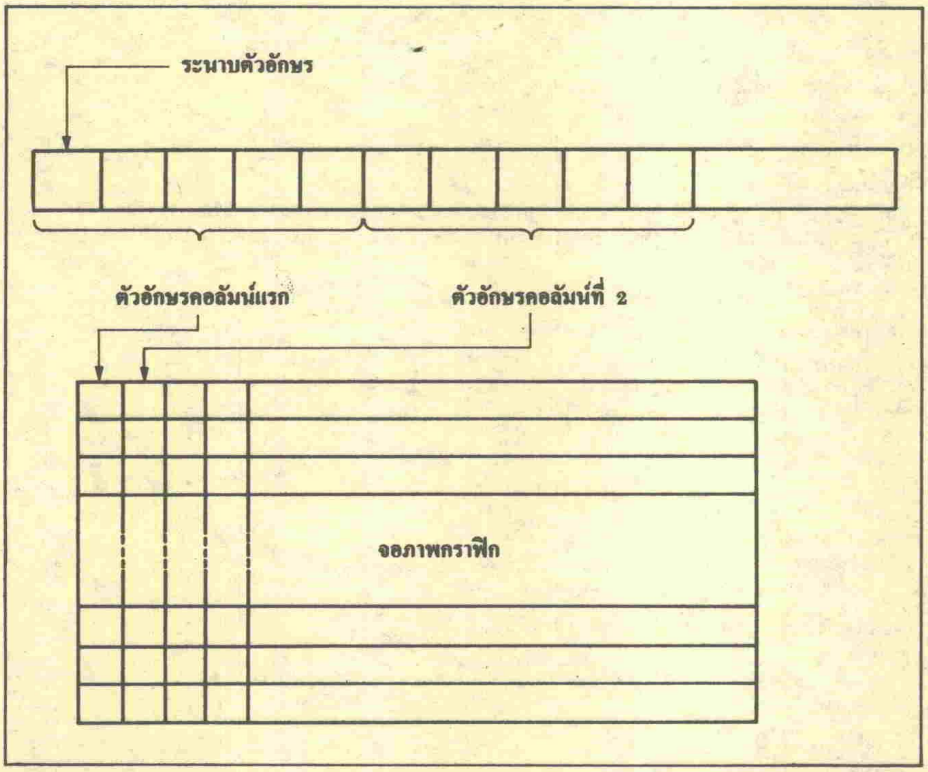
จอภาพลอจิคัล ตัว TKS เองจะใช้หน่วยความจำส่วนหนึ่ง(เรียกว่า Text plane หรือระนาบตัวอักษร) เป็นที่สำหรับเก็บเป็นรหัสตัวอักษรพร้อมทั้งแอตทริบิวต์ของตัวอักษรนั้น ซึ่งในหน่วยความจำส่วนนี้นั้น TKS จะมาอ่านดูรหัสตัวอักษรจากนั้นนำฟอนต์ (font) หรือภาพของตัวอักษรออกไปแสดงที่จอภาพฟิสิคัลตามตำแหน่ง



รูปที่ 3 จอภาพฟิสิคัลใน TKS แสดงขนาดของจอและส่วนประกอบของตัวอักษร 1 คอลัมน์



รูปที่ 4 จอภาพลอจิคัลหรือระนาบตัวอักษรและลักษณะการเก็บรหัสตัวอักษร



รูปที่ 5 ตำแหน่งตัวอักษรบนจอภาพแสดงผลหรือจอภาพฟิสิคัลกับการเก็บเรียงของระนาบตัวอักษรที่มีความสัมพันธ์กัน

ของตัวอักษรในระนาบตัวอักษร การจัดเรียงหน่วยความจำในส่วนระนาบ ตัวอักษรนี้ จะมีความสัมพันธ์กับตัวอักษรที่ปรากฏบนจอภาพฟิสิคัล ตัวอักษรหนึ่งคอลัมน์บนจอภาพฟิสิคัลนั้นในระนาบตัวอักษรจะใช้ 5 ไบต์ โดยไบต์แรกจะเก็บรหัสของวรรณยุกต์ ไบต์ที่สองเก็บรหัสของสระส่วนบน ไบต์ที่สามเป็นรหัสของพยัญชนะ ไบต์ที่สี่เป็นรหัสของสระส่วนล่าง คือพวกสระอุ สระอู และไบต์สุดท้ายคือไบต์ที่ห้าจะเป็นแอตทริบิวต์ของตัวอักษรทั้งสี่ระดับ ในคอลัมน์ถัดไปของจอภาพฟิสิคัลก็จะประกอบด้วย 5 ไบต์ถัดไป ในระนาบตัวอักษร ดังรูปที่ 4 และรูปที่ 5

ดังได้กล่าวมาแล้วว่าการแสดงบนจอภาพฟิสิคัล TKS จะทำการอ่านเอารหัสตัวอักษรที่จะต้องแสดงผลจากระนาบตัวอักษร จากนั้นจะดึงเอาฟอนต์ซึ่งจะมีขนาด 8×8 จุด ไปแสดงบนจอภาพฟิสิคัล นี้คือเหตุผลที่ทำให้จำนวนคอลัมน์และจำนวนบรรทัดบนจอภาพแบบเฮอร์คิวลิส และ CGA ไม่เท่ากันเนื่องจากความละเอียดของจอ จากลักษณะข้างต้นจะเห็นว่า ถึงแม้ TKS จะแสดงผลในลักษณะกราฟิกก็ตาม แต่สำหรับผู้ใช้ TKS จะเห็นการใช้งานในลักษณะ text โดยที่ผู้ใช้ไม่ต้องเข้ามายุ่งเกี่ยวกับการแสดงผลบนจอภาพ TKS จะมีฟังก์ชันการแสดงผลภาษาไทยได้ถูกต้อง ซึ่งตัว TKS จะจัดการเองทุกอย่างทั้งในจอภาพลอจิคัลและจอภาพฟิสิคัลโดยที่ผู้ใช้ไม่ต้องยุ่งเกี่ยวด้วยเลย ใน TKS มีฟังก์ชันให้ใช้มากมายดังรูปที่ 6

การเรียกใช้ระบบ TKS ในโปรแกรมที่ใช้งาน TKS จะต้อง include ไฟล์ 3 ไฟล์ต่อไปนี้คือ

1. Kernel.H

- เป็นส่วนที่ประกาศค่าตัวคงที่ที่ใช้ใน TKS และ Prototype ของฟังก์ชันรวมทั้งฟังก์ชันที่ใช้งานในระบบประมวลผลภาษาไทย

2. VDO_KNL.H

- เป็นส่วนประกาศค่าตัวคงที่ที่ใช้ระบบแสดงผลออกจอของ TKS และ Prototype ของฟังก์ชัน

3. KBD_KNL.H

- เช่นกันกับ 2 ไฟล์ข้างบน แต่เป็นของคีย์บอร์ด

```
ติดตั้งระบบ TKS
void    kernel(void)

- เปลี่ยนการแสดงเป็นกราฟิกโดย TKS
void    vdoset(m)

- นำตัวอักษรที่ตำแหน่ง x, y ในระนาบตัวอักษรมาแสดงผลบนจอกราฟิก
void    pokexych(x,y)
void    pokexycha(x,y,att)

- นำตัวอักษรที่ตำแหน่งเคอร์เซอร์ปัจจุบันในระนาบตัวอักษรมาแสดงผลบนจอกราฟิก
void    pokech()
void    pokecha(att)

- เลื่อนขึ้น, ลง ของหน้าต่าง
void    scrollup(x1,y1,x2,y2,n,att)
void    scrolldw(x1,y1,x2,y2,n,att)

- กำหนดตำแหน่ง, ลักษณะและการเปิด, ปิด เคอร์เซอร์
void    setcurpos(x,y)
void    setcurtype(a,b)
void    setcurstyle(a)
void    setcursor(n)

- อ่านตำแหน่ง,  ลักษณะและขนาดของเคอร์เซอร์
unsigned    getcurpos(void)
unsigned    getcurtype(void)
unsigned    getscrsize(void)

- เลือกการแสดงตัวอักษรกราฟิก
void    setgraphchar(unsigned)

- แสดงผลตัวอักษร  (พร้อมทั้งแอตทริบิวต์) ที่ตำแหน่งเคอร์เซอร์ปัจจุบัน
void    puttc(unsigned char);
void    putcha(unsigned char,int)

- แสดงผลตัวอักษร  (พร้อมทั้งแอตทริบิวต์) ที่ตำแหน่ง x, y จำนวน n ตัว
void    dispxyntc(int,int,int,unsigned char,unsigned char,unsigned char,unsigned char)
void    dispxyntca(int,int,int,unsigned char,unsigned char,unsigned char,unsigned char,int)
void    dispxynch(int,int,int,unsigned char)
void    dispxyncha(int,int,int,unsigned char,int)

- แสดงผลตัวอักษร  (พร้อมทั้งแอตทริบิวต์)  ที่ตำแหน่งเคอร์เซอร์ปัจจุบันจำนวน n ตัว
void    dispntc(int,unsigned char,unsigned char,unsigned char,unsigned char)
void    dispntca(int,unsigned char,unsigned char,unsigned char,unsigned char,int)
void    dispnch(int,unsigned char)
void    dispncha(int,unsigned char,int)


void    dispxystr(int,int,unsigned char far *)
void    dispxystra(int,int,unsigned char far *,int)
void    dispxynstr(int,int,int,unsigned char far *)
void    dispxynstra(int,int,int,unsigned char far *,int)

- แสดงผลข้อความตามลักษณะภาษาไทย (พร้อมทั้งแอตทริบิวต์)  ที่ตำแหน่งเคอร์เซอร์ปัจจุบัน
void    dispstr(unsigned char far *)
void    dispstra(unsigned char far *,int)
void    dispnstr(int,unsigned char far *)
void    dispnstra(int,unsigned char far *,int)

- กำหนดและอ่านโหมดการรับแป้นพิมพ์ (เป็นภาษาไทยหรืออังกฤษ)
int     getkbdmode(void)

- อ่านตัวอักษรจากแป้นพิมพ์
int     gettc(void)

- อ่านข้อความพร้อมทั้งแสดงผลภาษาไทยจากแป้นพิมพ์
int     getstr(unsigned char far *)
int     getnstr(int,unsigned char far *)
```

รูปที่ 6 ฟังก์ชันใน TKS

นอกจากนั้นในการใช้งานจริงๆ ก่อนอื่นจะต้องเรียกฟังก์ชัน KERNEL( ) ก่อนเสมอ และการแสดงผลออกจอกราฟิกจะเรียกฟังก์ชัน VDOSET(ON) ดังตัวอย่างการเรียกใช้ TKS ต่อไปนี้

```
# include <stdio.h>
  .
  .
  .
# include "Kernel.H"
# include "VDO-KNL.H"
# include "KBD-KNL.H"
main ()
{
  Kernel (); /*call this function
               before use TKS */
  VDO-Set (ON); /*Switch Graphic
                  mode display*/
}
```

โปรแกรมที่จะใช้ TKS จะต้องคอมพายล์ด้วยไมโครซอฟต์ซีโมเดลแบบ large และใส่ library option ลงไปด้วยดังนี้

C>CL-AL Test.C-Link Klib

โดยที่ Test.C คือโปรแกรมที่เรียกใช้ TKS

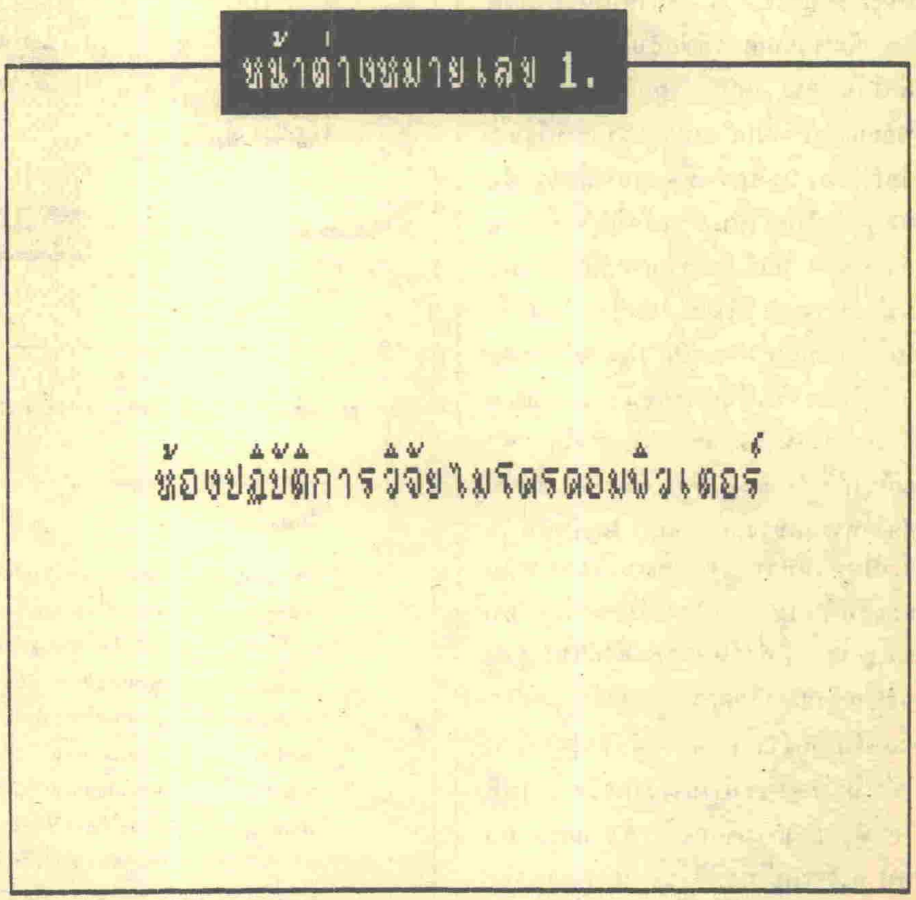
- AL คือการบอกโมเดลแบบ large
- link คือการบอกชื่อไลบรารี่ที่ต้องการลิงค์รวมด้วย

KLIB คือชื่อไลบรารี่ของ TKS

ระบบภาษาไทยระดับแก่นกับหน้าต่าง (TKS และ windows)

การสร้างหน้าต่าง(window) บน TKS เป็นการแสดงข้อความบนจอเพื่อความสวยงามและความสะดวก เรามักนิยมที่จะเปิดช่องหน้าต่างแล้วแสดงข้อความภายในช่องหน้าต่างที่เปิด การเปิดช่องหน้าต่างนั้นส่วนของจอภาพที่ถูกช่องหน้าต่างทับจะได้รับการเก็บรักษาไว้ และเมื่อช่องหน้าถูกปิด ส่วนของจอภาพที่ถูกปิดทับก็จะได้รับการนำกลับคืนดังเดิมก่อนหน้าที่จะเปิดช่องหน้าต่าง ในตัว TKS ในกรณีที่เราเปิดช่องหน้าต่างซึ่งช่องหน้าต่างจะเปลี่ยนแปลงขนาดได้ตามความต้องการแสดงข้อความบนจอภาพ และการเคลื่อนที่ของเคอร์เซอร์จะต้องอยู่ภายในขอบเขตของหน้าต่างที่เปิดอยู่ แต่ TKS ในรุ่นนี้ ยังไม่ได้มีฟังก์ชันส่วนการควบคุมดังกล่าว ใน TKS รุ่นต่อ ๆ ไป จะได้รับการพัฒนาและเพิ่มความสามารถในส่วนนี้เข้าไปด้วย

โครงสร้างข้อมูลของช่องหน้าต่าง การเปิดช่องหน้าต่างนั้นจะต้องกำหนดตำแหน่งของหน้าต่างโดยที่บอกเป็นจุดทะแยงมุมของช่องหน้าต่างคือ (X1, Y1) และ (X2, Y2) นอกจากนี้แล้วในโครงสร้างข้อมูลจะต้องมีส่วนที่เก็บตำแหน่งของเคอร์เซอร์ก่อนที่จะทำการเปิดหน้าต่างและส่วนที่จะเก็บข้อมูลบนจอภาพที่จะถูกทับ เมื่อภายหลังการปิดหน้าต่างจะได้คืนตำแหน่งเคอร์เซอร์และส่วนของจอภาพที่ถูกทับคืนดังเดิม และส่วนที่สำคัญอีกอันหนึ่งก็คือพอยน์เตอร์ที่จะชี้กลับไปยังโครงสร้างข้อมูลของช่องหน้าต่างที่ถูกเปิดขึ้นมาก่อน ลักษณะเช่นนี้การเปิดหน้าต่างจะคล้ายสแต็กซึ่งทำให้สามารถเปิดหน้าต่างได้จำนวนมากโดยขึ้นกับหน่วยความจำที่มีอยู่ หน้าต่างที่ถูกเปิดไปแล้วการปิดหน้าต่างจะปิดหน้าต่างที่ถูกเปิดหลังสุดแล้วไล่กลับเรื่อย ๆ ดังรูปที่ 7 หน้าต่างหมายเลข 1 ถูกเปิด รูปที่ 8 หน้าต่างหมายเลข 2 และ 3 ถูกเปิดตามมาภายหลัง และในรูปที่ 9 หน้าต่างหมาย

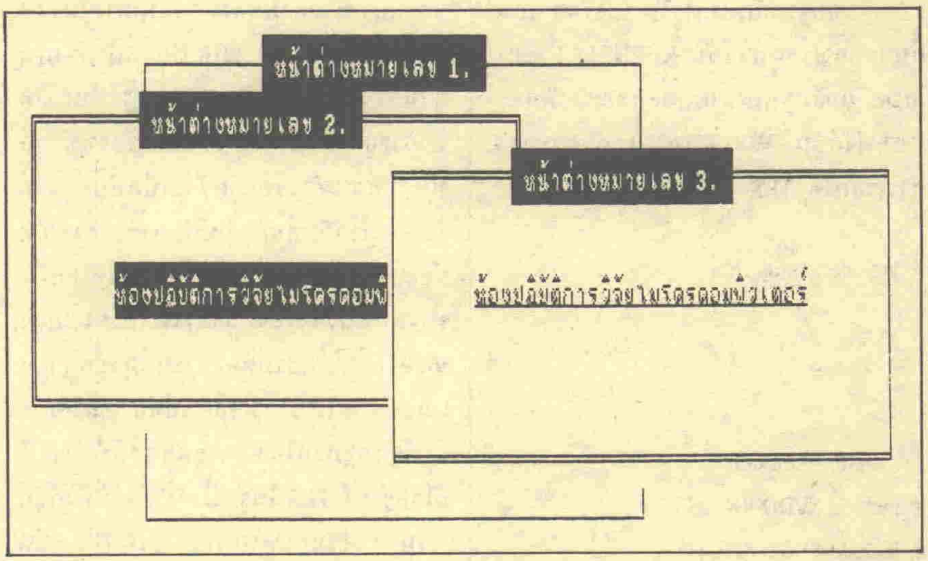


รูปที่ 7 หน้าต่างที่ 1 ถูกเปิดบน TKS

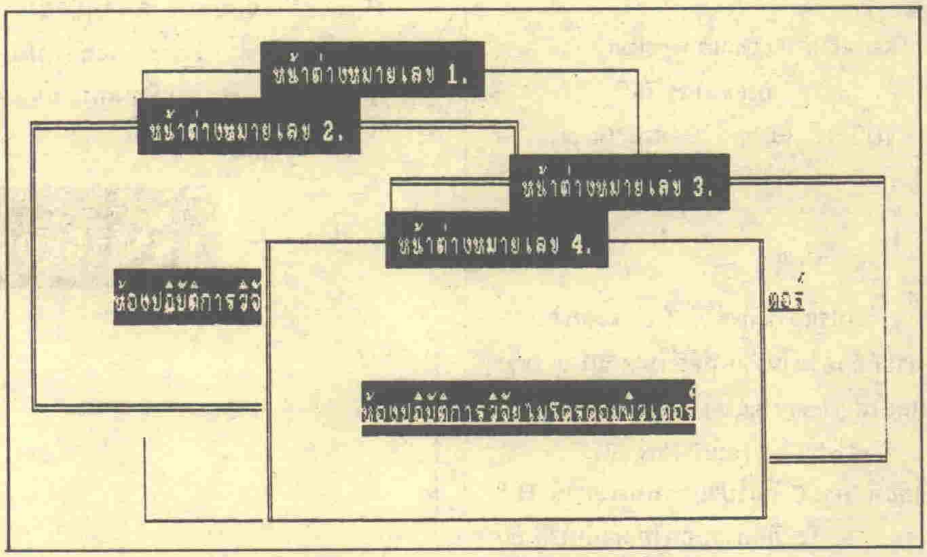
เลข 4 ถูกเปิดเป็นหน้าต่างสุดท้ายซึ่งอยู่บนสุดของหน้าต่างหมายเลข 3, 2 และ 1 การปิดหน้าต่างจะเริ่มปิดจากหน้าต่างหมายเลข 4 ก่อน แล้วจึงปิดหน้าต่างหมายเลข 3, 2 และ 1 ตามลำดับ โครงสร้างข้อมูลของหน้าต่างจะเป็นดังนี้

```
typedef struct WINDOWS
{   int x1,y1,x2,y2;
   int x,y;
   unsigned char id;
   struct WINDOWS *backlink;
   unsigned char far *screencontents;
} WINDOW_BLOCK, *WINDOW_LINK;
```

โดย X1 และ Y1 คือตำแหน่งมุมบนซ้ายของช่องหน้าต่าง X2 และ Y2 คือตำแหน่งมุมล่างขวาของช่องหน้าต่างที่จะเปิด X, Y คือตำแหน่งของเคอร์เซอร์ก่อนหน้าที่จะเปิดช่องหน้าต่างอันนี้ id คือหมายเลขประจำตัวของช่องหน้าต่าง back-link คือพอยน์เตอร์ที่ชี้กลับไปยังช่องหน้าต่างอันก่อน และ screencontents คือพอยน์เตอร์ที่ชี้ไปยังหน่วยความจำที่ใช้เก็บข้อความบนจอภาพที่จะถูกหน้าต่างทับ ข้อมูลที่เก็บจะเป็นส่วนข้อมูลในระนาบตัวอักษร ไม่เก็บจอภาพกราฟิก ดังนั้นหน่วยความจำที่ใช้ในส่วนนี้จะน้อยกว่าในกรณีเก็บเป็นจอกราฟิก ก่อนจะใช้ระบบหน้าต่างจะต้องเรียกฟังก์ชัน init-window ก่อนจะทำการเปิดและปิดหน้าต่างใด ๆ ตัวอยู่ในรูปที่ 11 และรูปที่ 12 (โปรแกรมทั้งสองจัดไว้ท้ายสุดของบทความนี้) โดยที่รูปที่ 11 จะเป็นส่วนหัวของโปรแกรม (KNLW.H) และรูปที่ 12 จะเป็นตัวโปรแกรมระบบหน้าต่าง ที่ฟังก์ชันแต่ละฟังก์ชันมีส่วนอธิบายรายละเอียดการใช้อยู่แล้ว แต่ก็จะขออธิบายคร่าว ๆ ถึงฟังก์ชันที่จะถูกเรียกใช้งานจากผู้ที่จะนำไปใช้งานจริง ในรูปที่ 10 ตัวอย่างการเรียกใช้ TKS และระบบหน้าต่างจะอยู่ในรูปที่ 13 และผลลัพธ์จะอยู่ในรูปที่ 7, 8 และ 9



รูปที่ 8 หน้าต่างที่ 1, 2 และ 3 ถูกเปิด



รูปที่ 9 หน้าต่างที่ 4 ถูกเปิดจะแสดงส่วนบนสุดของหน้าต่างที่ปิดก่อน

ชื่อฟังก์ชัน	การทำงาน
init_window( )	- จะกำหนดค่าเริ่มต้นให้กับตัวแปรที่ถูกใช้งานโดยระบบหน้าต่าง
sboxstyle( )	- บอกชนิดของกรอบหน้าต่างที่ถูกเปิด โดยที่กรอบหน้าต่างมี 4 ชนิดคือ SINGLE, DOUBLE, MIX1 และ MIX2
sboxattr( )	- กำหนดแอตทริบิวต์ของขอบหน้าต่างที่จะวาด
swinheader( )	- กำหนดข้อความที่ปรากฏอยู่ขอบบนหน้าต่าง
sheaderattr( )	- กำหนดแอตทริบิวต์ของข้อความส่วนหัวของขอบหน้าต่าง
swinattr( )	- กำหนดแอตทริบิวต์ของส่วนของพื้นหน้าต่างที่จะถูกเปิด
windowopen( )	- คำสั่งในการเปิดหน้าต่าง หน้าต่างที่จะเปิดสามารถตบแต่งได้ด้วยฟังก์ชันข้างบนที่กล่าวมาแล้ว
windowclose( )	- คำสั่งในการปิดหน้าต่าง

รูปที่ 10 ฟังก์ชันบางฟังก์ชันที่ผู้ใช้ควรทราบ

```
 1 /*
 2             EXAMPLE PROGRM USE WINDOWS ON TKS
 3 */
 4
 5 #include <stdio.h>
 6 #include "vdo_knl.h"
 7 #include "kernel.h"
 8 #include "kbd_knl.h"
 9 #include "knlw.h"
10
11 main()
12 {
13    kernel();
14    vdoset(ON);
15    init_window();
16
17    sboxstyle(SINGLE);
18    swinheader(" หน้าต่างหมายเลข 1. ");
19    if(windowopen(10,1,50,9))
20       dispxystra(17,5,"ห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์",ATTR_NORMAL);
21
22    getch();
23    sboxstyle(DOUBLE);
24    swinheader(" หน้าต่างหมายเลข 2. ");
25    if(windowopen(1,2,40,7))
26       dispxystra(8,5,"ห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์",ATTR_REVERSE);
27
28    getch();
29    sboxstyle(MIX1);
30    swinheader(" หน้าต่างหมายเลข 3. ");
31    if(windowopen(30,3,70,8))
32       dispxystra(37,5,"ห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์",ATTR_UNDERL);
33
34    getch();
35    sboxstyle(MIX2);
36    swinheader(" หน้าต่างหมายเลข 4. ");
37    if(windowopen(20,4,60,9))
38       dispxystra(28,7,"ห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์",ATTR_REV_UNL);
39
40    getch();
41    windowclose();
42
43    getch();
44    windowclose();
45
46    getch();
47    windowclose();
48
49    getch();
50    windowclose();
51 }
```

รูปที่ 13 โปรแกรมตัวอย่างการเรียกใช้ TKS และระบบหน้าต่าง

ระบบภาษาไทยระดับแก่นกับระบบเมนู

ระบบเมนูบน TKS ที่สร้างขึ้นเป็นลักษณะของเมนูบาร์ หลักการทำงานของเมนูบาร์ก็คือ หลังจากที่แสดงหัวข้อหรือข้อความที่จะให้เลือกแล้ว จะรอรับการกดคีย์และตรวจคีย์ที่ถูกกด ถ้าเป็นคีย์ที่ใช้เลื่อนเคอร์เซอร์ (ARROW LEFT, RIGHT, DOWN, UP, PAGE-UP, PAGE-DOWN, HOME, END) จะเลื่อนบาร์ (ซึ่งสามารถกำหนดแอตทริบิวต์ได้) ไปมาตามหัวข้อหรือข้อความในตำแหน่งต่าง ๆ ถ้าเป็นการกดคีย์รีเทิร์นจะหมายถึงการเลือกหัวข้อหรือข้อความนั้น ๆ ที่มีบาร์อยู่ นอกจากนั้นยังสามารถกำหนดคีย์พิเศษ (เรียกว่า Hot Key) สำหรับการเลือกหัวข้อโดยตรงได้ด้วย

เมื่อนำเมนูบาร์รวมเข้ากับหน้าต่างก็จะได้เมนูแบบพูลดาวน์หรือป๊อปอัพ (pull-down, pop-up) คือการเปิดช่องหน้าต่างก่อนเพื่อบรรจุเมนูบาร์ลงไป หลังจากทำงานเรียบร้อยแล้วก็ทำการปิดช่องหน้าต่าง ระบบเมนูแบบพูลดาวน์หรือป๊อปอัพเป็นที่นิยมใช้กันมากในซอฟต์แวร์ทั่ว ๆ ไป เช่น เทอร์โบซี, เทอร์โบปาสคาลเวอร์ชัน 4, ไมโครซอฟต์เอ็กเซล ฯลฯ

ต่อไปเราจะมาดูโครงสร้างข้อมูลเมนูบาร์กัน ระบบเมนูบาร์นี้สามารถที่จะสร้างให้หัวข้อหรือข้อความให้เลือกมีจำนวนแถวหลายแถว และหลายหลักก็ได้ เช่น ในรูปที่ 14 มีหัวข้อเลือก 2 แถว และ 3 หลัก และในแถวสุดท้ายจะมีหัวข้อเลือกครบหรือไม่ครบหลักก็ได้ โครงสร้างข้อมูลของเมนูบาร์จะเป็นดังรูปที่ 15 โดยที่ top_x และ top_y เป็นตำแหน่งเริ่มต้นของข้อความแรกที่จะปรากฏในเมนูบาร์, num_row และ num_col เป็นจำนวนแถว และจำนวนหลักของเมนูบาร์ ส่วน total เป็นจำนวนหัวข้อเลือกทั้งหมด ในกรณี

ที่แถวสุดท้ายมีหัวข้อเลือกเท่ากับจำนวนหลัก ค่า total จะมีค่าเท่ากับ num_row คูณด้วย num_col แต่ถ้าในแถวสุดท้ายมีหัวข้อเลือกไม่เต็ม total ก็จะมีค่าเท่ากับหัวข้อเลือกทั้งหมดที่มีอยู่, width คือความกว้างของบาร์ และ space คือระยะห่างระหว่างหัวข้อเลือกในแนวแถวแต่ละหัวข้อ, attr_c คือแอตทริบิวต์ของตัวอักษรหัวข้อเลือกของเมนู และ attr_b คือแอตทริบิวต์ของบาร์ แอตทริบิวต์ใน TKS นี้จะมีอยู่ 4 อย่างคือ แบบ normal, reverse, under_line, และ reverse ผสม under_line hkey คือ hot_key หรือคีย์ที่ยอมให้กด โดย hot_key จะมีจำนวนคีย์เท่ากับหัวข้อเลือกและเรียงตามลำดับหัวข้อเรื่อง ส่วนสุดท้ายในโครงสร้างคือ item จะเป็นหัวข้อเลือกที่จะแสดงบนเมนูบาร์ ในการสร้างเมนูบาร์จะต้องเตรียมสร้างหัวข้อเลือกเสียก่อน กำหนดตำแหน่งความกว้างของบาร์, ระยะห่าง และข้อมูลทั้งหมดที่ต้องการในเมนูตามที่กล่าวไปแล้ว การเตรียมหัวข้อเลือกนั้นทำได้ง่ายโดยการสร้างตัวแปรที่มี Type เป็น MENU_MSG (MENU_MSG ถูกประกาศไว้ในไฟล์ KNLW.H) ตัวอย่างเช่น

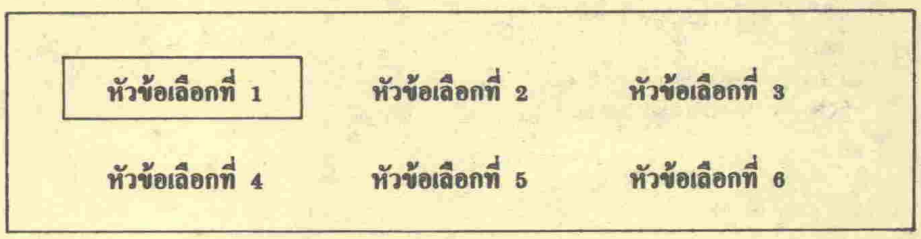
```
MENU-MSG menu-choice =

    {"หัวข้อเลือกที่ 1",
     "หัวข้อเลือกที่ 2",
     .
     .
     .
     .
     "หัวข้อเลือกที่ N",
    };
```

ส่วนตัวเมนูเองนั้นก็ต้องสร้างตัวแปรที่มีโครงสร้างข้อมูลตามที่กล่าวมาแล้ว การประกาศตัวแปรทำได้ดังนี้

```
MENU menu 1, menu 2;
```

จากนั้นใช้ฟังก์ชันสร้างเมนูบาร์ (make_menu) ดังนี้

```
make_menu (33, 3,/*ตำแหน่งเริ่มต้น เมนู*/
1,7,7, /*จำนวนแถว, จำนวนหลักและจำนวน
หัวข้อเลือกทั้งหมดของเมนู */
```



รูปที่ 14 เมนูบาร์ที่หัวข้อเลือกทั้งหมด 6 หัวข้อ 2 แถว และ 3 หลัก

```
4.7. /*ระยะห่างระหว่างหัวข้อและความกว้าง
บาร์*/
ATTR-REVERSE,ATTR-NORMAL/*แอต
ทริบิวต์ของอักษรและของบาร์*/
menu-choice, /*หัวข้อเลือก*/
& menu ); /*แอดเดรสของตัวแปร menu
*/
```

make-menu จะตรวจสอบขนาดของเมนูและตำแหน่งว่าอยู่ขอบเขตของจอหรือไม่ ถ้าไม่สามารถบรรจุได้จะส่งค่า 0 กลับมา ฟังก์ชันเมนูบาร์มีรูปแบบการใช้คือ

```
menubar (& key, & choice, & menu);
```

หลังจากเรียกฟังก์ชัน menubar แล้ว จะปรากฏหัวข้อเลือกและบาร์บนจอ แล้วรอรับการกดคีย์ตัวแปร key และ choice ที่ส่งผ่านให้ฟังก์ชัน menubar จะส่งค่ากลับคืนมา คีย์ที่ถูกกดส่งคืนทางตัวแปร key (คีย์ที่ยอมให้กดในฟังก์ชัน menubar นอกจากคีย์เลื่อนเคอร์เซอร์แล้วยังรับ RETURN และ ESC ด้วย) ส่วนหัวข้อที่เลือกถูกส่งกลับทางตัวแปร choice ตัวอย่างโปรแกรมที่ใช้เมนูบาร์จะอยู่ในรูปที่ 18 ส่วนรูปที่ 16, 17 เป็นผลลัพธ์ของโปรแกรม

ตัวโปรแกรมระบบหน้าต่างและเมนูบาร์ซึ่งอยู่ในรูปที่ 12 ชื่อว่า KNLW.C รวมทั้ง Header File KNLW.H ในรูปที่ 11 KNLW.C จะถูกคอมพายล์เป็นไฟล์ออฟเจ็ค (.OBJ) แล้วในซอฟต์เมล์ จะมีไฟล์ดังต่อไปนี้คือ

1. ส่วนของ TKS
   - KERNEL.H
   - VDO_KNL.H
   - KBD_KNL.H
   - TKS.LIB

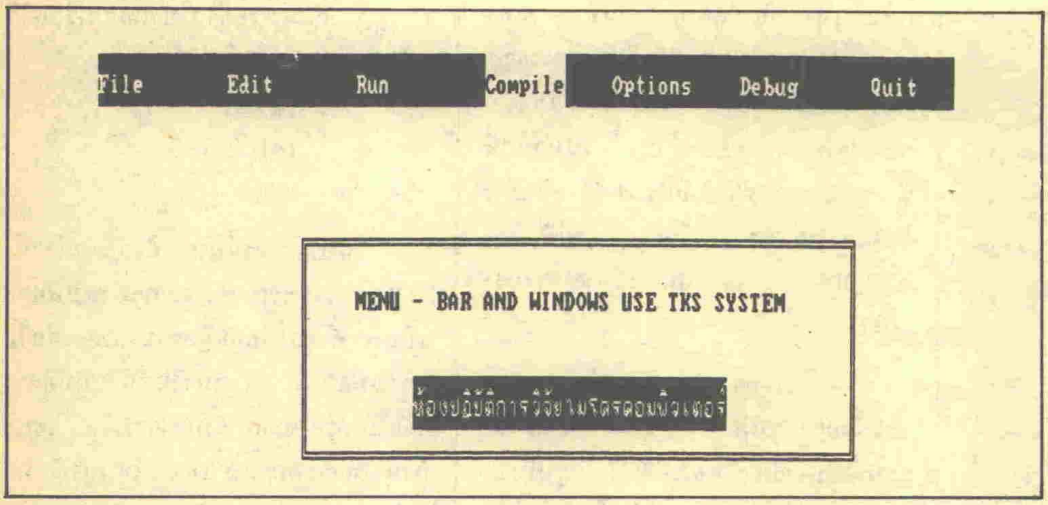
2. ส่วนของหน้าต่างและเมนู
   - KNLW.H
   - KNLW.C
   - KNLW.OBJ

3. ส่วนตัวอย่างโปรแกรมเรียกใช้หน้าต่างและเมนู
   - TKW.C
   - TKB.C

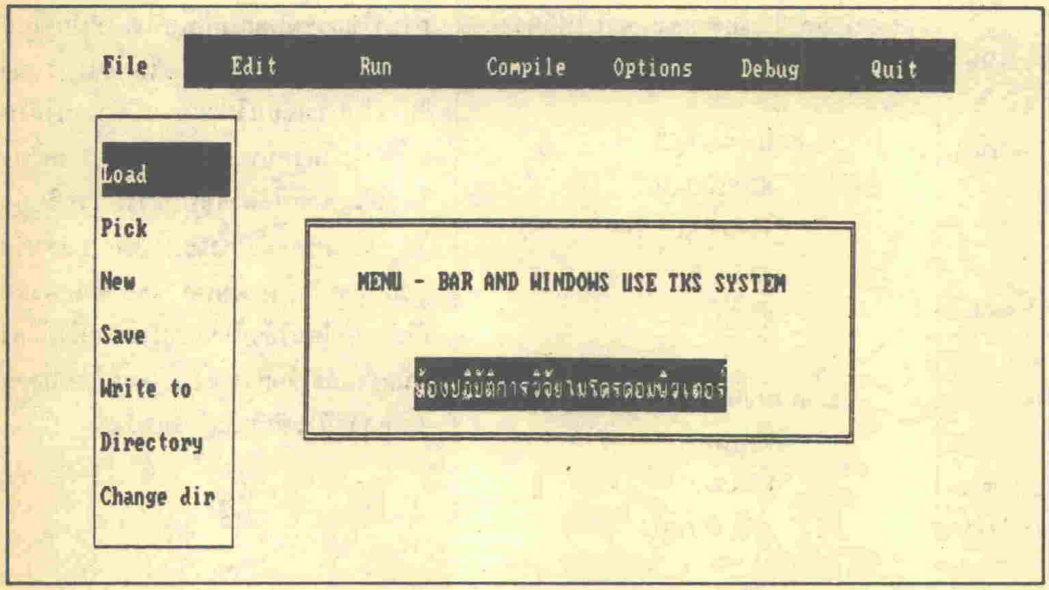
ในบทความนี้นั้น ยังมิได้กล่าวถึง TKS ในรายละเอียดด้านเครื่องพิมพ์และแป้นพิมพ์ ขอให้คอยติดตามในตอนต่อไปของบทความ ทางทีมงานห้องปฏิบัติการวิจัยไมโครฯ จะพยายามทำการพัฒนาระบบภาษาไทยระดับแก่น TKS ให้สมบูรณ์ วัตถุประสงค์ของการวิจัยและพัฒนาระบบภาษาไทยระดับแก่นก็เพื่อที่จะให้นักเขียนโปรแกรมที่เกี่ยวกับภาษาไทยสามารถนำไปใช้ในการเขียนโปรแกรมที่ทำงานได้ในเครื่องไมโครคอมพิวเตอร์ทุก ๆ เครื่อง โดยที่ไม่ต้องพึ่งพาระบบฮาร์ดแวร์อีกต่อไป ในอนาคตนั้นคงจะมีโปรแกรมสำเร็จรูปสเปรดชีต, ดาต้าเบส ฯลฯ ที่แสดงผลเป็นภาษาไทยได้อย่างสมบูรณ์โดยที่ไม่ต้องขึ้นตรงต่อฮาร์ดแวร์ ซึ่งทำโดยฝีมือของคนไทยเอง มิใช่ที่เป็นอยู่ในทุกวันนี้

```
typedef unsigned char * MENU_MSG[MAX_ITEMS];

typedef struct
{
        int top_x;                  /* Start x of menu-bar              */
        int top_y;                  /* Start y of menu-bar              */
        int num_row;                /* Number of row in menu            */
        int num_col;                /* Number of column in menu         */
        int total;                  /* Total number of menu choice      */
        int space;                  /* Horizontal space of each choice  */
        int width;                  /* Width of light bar               */
        int attr_c;                 /* Attribute of choice              */
        int attr_b;                 /* Attribute of bar                 */
        unsigned char *hkey;        /* Hot key                          */
        unsigned char **item;       /* Item chioce                      */
} MENU;
```

รูปที่ 15 โครงสร้างข้อมูลของเมนูบาร์



รูปที่ 16 ผลของตัวอย่างโปรแกรมใช้เมนูบาร์



รูปที่ 17 เมื่อกดคีย์ "F"

รูปที่ 18 โปรแกรมตัวอย่างการเรียกใช้เมนูบาร์

```
 1 /*
 2              EXAMPLE PROGRAM USE MENU - BAR ON TKS
 3 */
 4
 5
 6 #include <stdio.h>
 7 #include <ctype.h>
 8 #include "kernel.h"
 9 #include "vdo_knl.h"
10 #include "kbd_knl.h"
11 #include "knlw.h"
12
13
14 MENU menu_main,menu1,
15      menu4,menu5,menu6;
16
17 MENU_MSG menu_choice  = { "File", "Edit", "Run", "Compile",
```

```
18                          "Options", "Debug", "Quit" };
19
20 MENU_MSG menu_choice1 = { "Load","Pick","New","Save",
21                          "Write to","Directory","Change dir" };
22
23 MENU_MSG menu_choice4 = { "Compile to Obj","Make EXE file",
24                          "Link EXE file","Build all","Primary C file" };
25
26 MENU_MSG menu_choice5 = { "Compiler","Linker","Enviroment",
27                          "Directory","Arguments" };
28
29 MENU_MSG menu_choice6 = { "Track Massages","Clear Massages",
30                          "Keep Massages" };
31
32 void command1()
33 {
34     int key=0,choice=1;
35
36      key=0;
37      swinheader("\0");
38      sboxstyle(SINGLE);
39      windowopen(2,2,14,10);
40      while(key!=27)
41           menubar(&key,&choice,&menu1);
42      windowclose();
43 }
44
45 void command4()
46 {
47     int key=0,choice=1;
48
49      key=0;
50      swinheader("\0");
51      sboxstyle(SINGLE);
52      windowopen(32,2,48,8);
53      while(key!=27)
54           menubar(&key,&choice,&menu4);
55      windowclose();
56 }
57
58
59 void command5()
60 {
61     int key=0,choice=1;
62
63      key=0;
64      swinheader("\0");
65      sboxstyle(SINGLE);
66      windowopen(46,2,58,8);
67      while(key!=27)
68           menubar(&key,&choice,&menu5);
69      windowclose();
70 }
71
72
73 void command6()
74 {
75     int key=0,choice=1;
76
77      key=0;
78      swinheader("\0");
79      sboxstyle(SINGLE);
80      windowopen(56,2,72,6);
```

```
     while(key!=27)
          menubar(&key,&choice,&menu6);
     windowclose();
}

main()
{
   int key=0,choice=1;
   int key1=0,choice1=1;

   kernel();
   vdoset(ON);
   init_window();

   make_menu(3,1,                          /* X,Y postion of menu-bar */
             1,7,7,                        /* Number of row ,column and total choice */
             4,7,                          /* Space between choice and width of bar */
             ATTR_REVERSE,ATTR_NORMAL,     /* Attribute of choice and attribute of bar */
             "FERCODQ\0",                  /* Define hot keys */
             menu_choice,                  /* Menu choices    */
             &menu_main);                  /* Menu structure  */

   make_menu(3,3,7,1,7,0,11,ATTR_NORMAL,ATTR_REVERSE,
             "LPNSWDC\0",menu_choice1,&menu1);

   make_menu(33,3,5,1,5,0,15,ATTR_NORMAL,ATTR_REVERSE,
             "CMLBP\0",menu_choice4,&menu4);

   make_menu(47,3,5,1,5,0,11,ATTR_NORMAL,ATTR_REVERSE,
             "CLEDA\0",menu_choice5,&menu5);

   make_menu(57,3,3,1,3,0,15,ATTR_NORMAL,ATTR_REVERSE,
             "TCK\0",menu_choice6,&menu6);

   sboxstyle(DOUBLE);
   windowopen(20,4,67,8);
   dispxystra(25,5,"MENU - BAR AND WINDOWS USE TKS SYSTEM",ATTR_NORMAL);
   dispxystra(30,7,"ห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์",ATTR_REVERSE);
   getch();

   while(key!=27 && choice!=7)
      {
        menubar(&key,&choice,&menu_main);
        if(key!=27)
           {
             switch(choice)
                {
                  case 1 : command1();
                           break;
                  case 4 : command4();
                           break;
                  case 5 : command5();
                           break;
                  case 6 : command6();
                           break;
                }
           }
      }
   windowclose();
}
```

รูปที่ 11 เฮดเดอร์ของโปรแกรมระบบหน้าต่าง TKS

```
/*
                WINDOW AND MENU-BAR ON THAI KERNEL
                    create date : 11/10/1988
*/

#define BLANK_KEY     32
#define CR            13
#define ESC           27
#define TAB_KEY        9
#define SHIFT_TAB    271
#define CTRL_A         1
#define CTRL_C         3
#define CTRL_D         4
#define CTRL_E         5
#define CTRL_F         6
#define CTRL_G         7
#define CTRL_I        10
#define CTRL_K        11
#define CTRL_L        12
#define CTRL_N        14
#define CTRL_O        15
#define CTRL_P        16
#define CTRL_Q        17
#define CTRL_R        18
#define CTRL_S        19
#define CTRL_T        20
#define CTRL_V        22
#define CTRL_W        23
#define CTRL_X        24
#define CTRL_Y        25
#define CTRL_Z        26
#define ATL_1        376
#define ATL_2        377
#define ATL_3        378
#define ATL_4        379
#define ATL_5        380
#define ATL_6        381
#define ATL_7        382
#define ATL_8        383
#define ATL_9        384
#define ATL_0        385
#define HOMEKEY      327
#define ENDKEY       335
#define UPKEY        328
#define DOWNKEY      336
#define PGUPKEY      329
#define PGDNKEY      337
#define LEFTKEY      331
#define CENTERKEY    332
#define INSKEY       338
#define RIGHTKEY     333
#define DELKEY       339
#define CTRL_LEFT    371
#define CTRL_RIGHT   372
#define CTRL_PGUP    388
#define CTRL_PGDN    374
#define CTRL_HOME    378
#define CTRL_END     373
#define CTRL_UP      397
#define CTRL_DOWN    401
#define CTRL_CENTER  399
#define F1           315
#define F2           316
#define F3           317
#define F4           318
#define F5           319
#define F6           320
#define F7           321
#define F8           322
#define F9           323
#define F10          324
#define CTRL_F1      350
#define CTRL_F2      351
#define CTRL_F3      352
#define CTRL_F4      353
#define CTRL_F5      354
#define CTRL_F6      355
#define CTRL_F7      356
#define CTRL_F8      357
#define CTRL_F9      358
#define CTRL_F10     359
#define SHIFT_F1     340
#define SHIFT_F2     341
#define SHIFT_F3     342
#define SHIFT_F4     343
#define SHIFT_F5     344
#define SHIFT_F6     345
#define SHIFT_F7     346
#define SHIFT_F8     347
#define SHIFT_F9     348
#define SHIFT_F10    349
#define ALT_F1       360
#define ALT_F2       361
#define ALT_F3       362
#define ALT_F4       363
#define ALT_F5       364
#define ALT_F6       365
#define ALT_F7       366
#define ALT_F8       367
#define ALT_F9       368
#define ALT_F10      369
#define MAX_ITEMS     50
#define SINGLE         0
#define DOUBLE         1
#define MIX1           2
#define MIX2           3

/*-------------------------------------------------------------------*/

typedef struct WINDOWS
{       int x1,y1,x2,y2;
    int x,y;
    unsigned char id;
    struct WINDOWS *backlink;
```

```
115    unsigned char far *screencontents;
116 } WINDOW_BLOCK, *WINDOW_LINK;
117
118
119 typedef unsigned char * MENU_MSG[MAX_ITEMS];
120
121 typedef struct
122 {
123         int top_x;                  /* Start x of menu-bar            */
124         int top_y;                  /* Start y of menu-bar            */
125         int num_row;                /* Number of row in menu          */
126         int num_col;                /* Number of column in menu       */
127         int total;                  /* Total number of menu choice    */
128         int space;                  /* Horizontal space of each choice */
129         int width;                  /* Width of light bar             */
130         int attr_c;                 /* Attribute of choice            */
131         int attr_b;                 /* Attribute of bar               */
132         unsigned char *hkey;        /* Hot key                        */
133         unsigned char **item;       /* Item chioce                    */
134 } MENU;
135
136 /*-------------------------------------------------------------------*/
137
138 void            init_windown  (void);
139 void            sheaderattr   (unsigned char attrib);
140 void            sboxstyle     (unsigned char stlye);
141 void            sboxattr      (unsigned char attrib);
142 void            swinheader    (unsigned char *headerstring);
143 void            swinattr      (unsigned char attrib);
144 int             windowopen    (int x1,int y1,int x2,int y2);
145 void            windowclose   (void);
146 void            windowbox     (int x1,int y1,int x2,int y2);
147 unsigned int    getkey        (void);
148 void            putxyscr      (unsigned char far * sor,int x,int y,int n);
149 void            readxyscr     (unsigned char far * dest,int x,int y,int n);
150 void            menubar       (int *key,int *choice,MENU * mb);
151 void            disp_bar      (int x,int y,int width,int attr);
152 void            disp_menu     (MENU *mb);
153
```

รูปที่ 12 ตัวโปรแกรมระบบหน้าต่าง TKS

```
  1 /*
  2                       WINDOWS AND MENU - BAR
  3 */
  4
  5 #include <bios.h>
  6 #include <ctype.h>
  7 #include <malloc.h>
  8 #include <memory.h>
  9 #include <stdio.h>
 10 #include "kernel.h"
 11 #include "vdo_knl.h"
 12 #include "kbd_knl.h"
 13 #include "knlw.h"
 14
 15 extern  unsigned char  near * scrbuff;
 16 extern  unsigned char  scr_xsize;
 17 extern  unsigned char  scr_ysize;
 18 extern  unsigned char  cur_x;
 19 extern  unsigned char  cur_y;
 20
 21 const int type_of_box[4][8]={ { 196,179,179,196,218,191,192,217 },
 22                                { 205,186,186,205,201,187,200,188 },
 23                                { 205,179,179,205,213,184,212,190 },
 24                                { 196,186,186,196,214,183,211,189 } };
 25
 26 const int top=0,left=1,right=2,bottom=3;
 27 const int upleft=4,upright=5,loleft=6,loright=7;
 28 WINDOW_LINK    activewindow;
 29 int            fixedsize;
 30 unsigned char  windowcount;
 31 int            boxstyle;
 32 unsigned char  *header_of_window;
 33 unsigned char  attr_of_window;
 34 unsigned char  attr_of_header;
 35 unsigned char  attr_of_box;
 36
 37 /*-------------------------------------------------------------------
 38
 39 init_window     initialize window variable
 40
 41 Usage           void  initwin(void);
 42
 43 Description     To initialize window variable before window function
 44                 must call this function first time.
 45
 46 Return value    no value to return
 47
 48 */
 49
 50
 51 void init_window(void)
 52 {
 53      activewindow=NULL;
 54      windowcount=0;
 55      fixedsize=sizeof(WINDOW_BLOCK);
 56      boxstyle=SINGLE;
 57      *header_of_window= '\0';
 58      attr_of_window  = ATTR_NORMAL;
 59      attr_of_header  = ATTR_REVERSE;
 60      attr_of_box     = ATTR_NORMAL;
 61 }
 62
 63 /*-------------------------------------------------------------------
 64
 65 windowbox       Draw a window box
 66
 67 Usage           void  windowbox (int x1,int y1,int x2,int y2);
 68
 69 Description     This function write box on a screen at coordinate
 70                 (x1,y1)
```

```
                    +----------+
                    |          |
                    |          |
                    +----------+
                              (x2,y2)

Return value    no value return

*/

void windowbox(x1,y1,x2,y2)
int x1,x2,y1,y2;
{
int   x,y;

      setgraphchar(ON);
      scrollup(x1,y1,x2,y2,0,attr_of_window);
      dispxyncha(x1,y1,1,type_of_box[boxstyle][upleft],
                 attr_of_box|ATTR_GRAPHIC);
      dispxyncha(x1+1,y1,(x2-x1)-1,type_of_box[boxstyle][top],
                 attr_of_box|ATTR_GRAPHIC);
      dispxyncha(x2,y1,1,type_of_box[boxstyle][upright],
                 attr_of_box|ATTR_GRAPHIC);
      for(y=y1+1;y<=y2-1;y++)
          {
            dispxyncha(x1,y,1,type_of_box[boxstyle][left],
                       attr_of_box|ATTR_GRAPHIC);
            dispxyncha(x2,y,1,type_of_box[boxstyle][right],
                       attr_of_box|ATTR_GRAPHIC);
          }
      dispxyncha(x1,y2,1,type_of_box[boxstyle][loleft],
                 attr_of_box|ATTR_GRAPHIC);
      dispxyncha(x1+1,y2,(x2-x1)-1,type_of_box[boxstyle][bottom],
                 attr_of_box|ATTR_GRAPHIC);
      dispxyncha(x2,y2,1,type_of_box[boxstyle][loright],
                 attr_of_box|ATTR_GRAPHIC);
      setgraphchar(OFF);
      if(strlen(header_of_window)!=0)
         dispxystra((x1+x2-strlen(header_of_window)) >> 1,y1,
                    header_of_window,attr_of_header);
}

/*-------------------------------------------------------------------

windowopen      Open a window

Usage           unsigned char  windowopen(int x1,int y1,int x2,int y2);

Description     Open window at (x1,y1)-(x2,y2) and save the screen to
                buffer ,when close window it restore screen content .
                The window that Open last will close first (like a stack).
                To set box stlye , box attribute ,window attribute ,
                header window, attribute of header see the fuction below

Return value    return 0 when cann't open window

*/
```

```
int  windowopen(x1,y1,x2,y2)
int x1,x2,y1,y2;
{
      WINDOW_LINK block;
      unsigned char far *scrb;
      int linelen,scrsize,y;

      linelen=(x2-x1+1);
      scrsize=linelen*(y2-y1+1)*5;
      if(x2>scr_xsize || y2>scr_ysize ||  (x2-x1)<2 || (y2-y1)<2)
        return(0);
      else
          {
            block=(WINDOW_BLOCK *)(malloc(fixedsize));
            scrb =(unsigned char far *)(malloc(scrsize));
          }
      if(block && scrb)
          {
            block->x1=x1;
            block->x2=x2;
            block->y1=y1;
            block->y2=y2;
            block->x=cur_x;
            block->y=cur_y;
            block->backlink=activewindow;
            activewindow=block;
            windowcount++;
            block->id=windowcount;
            block->screencontents=scrb;
            for(y=y1;y<=y2;y++)
                readxyscr((scrb + linelen*(y-y1)*5),x1,y,linelen);
            windowbox(x1,y1,x2,y2);
            return(1);
          }
      else
           return(0);
}

/*-------------------------------------------------------------------

windowcolse     Close window

Usage           void  windowclose (void);

Description     Colse a window that Open the last, and restore screen
                content.

Return value    no return value

*/

void windowclose(void)
{
      WINDOW_LINK block;
      int linelen,y;

      if(!activewindow==NULL)
```

```
        {
          block=activewindow;
          linelen=(block->x2)-(block->x1)+1;
          windowcount--;
          setgraphchar(ON);
          for(y=block->y1;y<=block->y2;y++)
              {
                putxyscr((block->screencontents+linelen*(y - block->y1)*5),
                         block->x1,y,linelen);
                pokexynstr(block->x1,y,linelen);
              }
          setgraphchar(OFF);
          activewindow=block->backlink;
          setcurpos(block->x,block->y);
          free(block->screencontents);
          free(block);
        }
}


/*-------------------------------------------------------------------

sboxstyle       Set type of box

Usage           void  sboxstyle(unsigned char stlye);

Description     Set tpye of box ,style type is SINGLE,DOUBLE,
                MIX1 and MIX2

Return value    no return value

*/

void sboxstyle(unsigned char stlye)
{
   boxstyle = stlye;
}

/*-------------------------------------------------------------------

sheader         Set header window

Usage           void  swinheader(unsigned char *headerstring);

Description     Set the header of window with  any string massage ,when
                open the window the header massge will write at the head
                of window.

Return value    no return value

*/

void swinheader(unsigned char *headerstring)
{
     header_of_window = headerstring;
}

/*-------------------------------------------------------------------

sheaderattr     Set attribute of header window

Usage           void sheaderattr(unsigned char attrib);

Description     Set header attribute  of window

Return value    no return value

*/

void sheaderattr(unsigned char attrib)
{
     attr_of_header = attrib;
}

/*-------------------------------------------------------------------

swinattr        Set window attribute

Usage           void  swinattr(unsigned char attrib);

Description     Set attribute of window

Return value    no return value

*/

void swinattr(unsigned char attrib)
{
     attr_of_window = attrib;
}

/*-------------------------------------------------------------------

sboxattr        Set attribute of box

Usage           void sboxattr (unsigned char attrib);

Description     Set box attribute of window

Return value    no return value

*/

void sboxattr(unsigned char attrib)
{
     attr_of_box = attrib;
}

/*-------------------------------------------------------------------*/

void readxyscr(unsigned char far * dest,int x,int y,int n)
{
     memmove(dest,(scrbuff+((y-1)*scr_xsize+(x-1))*5),n*5);
}
```

```
300
301 /*-------------------------------------------------------------------*/
302
303 void putxyscr(unsigned char far * sor,int x,int y,int n)
304 {
305      memmove((scrbuff+((y-1)*scr_xsize+(x-1))*5),sor,n*5);
306 }
307
308 /*-------------------------------------------------------------------
309
310 getkey          Get key board
311
312 Usage           unsigned int getkey (void);
313
314 Description     Wait keyborad hit
315
316 Return value    return character key hit or extension key
317
318 */
319
320 unsigned int getkey()
321 {
322      register unsigned int c;
323
324      c=_bios_keybrd(_KEYBRD_READ);
325      if( c & 0xff )
326        return(c & 0xff);
327      else
328         return((c >> 8) + 256);
329 }
330
331 /*-------------------------------------------------------------------
332 make_menu       make MENU structure
333
334 usage           int manke_menu(x,y,num_rowm,num_colm,totalm,spacem,widthb,
335                                attr_cm,attr_bm,hot_key,itemmenu,menu);
336
337 Discrition      Make a structure menu bar , x y are start postion of
338                 manu, num_row num_col are number of row and column of
339                 item choice ,totalm is total item choice in menu bar,
340
341
342 Return value    if it can create  menu return 1
343                 if not create menu will return 0
344 */
345
346 int  make_menu(x,y,num_rowm,num_colm,totalm,spacem,widthb,
347               attr_cm,attr_bm,hot_key,itemmenu,menu)
348
349 int x,y;                /* Menu position start x,y */
350 int num_rowm,num_colm;  /* Number of row and column item choice  */
351 int totalm;             /* Total item choices of menu-bar */
352 int spacem,widthb;      /* Space between item choice and width of bar */
353 int attr_cm,attr_bm;    /* Attribute of choice charactor and attribute of bar */
354 unsigned char * hot_key;/* Hot key of menu choice */
355 MENU_MSG itemmenu;      /*Array of  pointer to string */
356 MENU * menu;            /* Pointer to MENU structure */
```

▶

```
357
358 {
359      if((x>0 && x<=scr_xsize) && (y>0 && y<=scr_ysize) &&
360         ((x+num_colm*(widthb+spacem)) <=scr_xsize)    &&
361         ((y+num_rowm-1) <=scr_ysize))
362        {
363          menu->top_x=x;
364          menu->top_y=y;
365          menu->num_row=num_rowm;
366          menu->num_col=num_colm;
367          menu->total=totalm;
368          menu->space=spacem;
369          menu->width=widthb;
370          menu->attr_c=attr_cm;
371          menu->attr_b=attr_bm;
372          menu->hkey=hot_key;
373          menu->item=(unsigned char  **)itemmenu;
374          return(1);
375        }
376      else
377         return(0);
378 }
379
380 /*-------------------------------------------------------------------*/
381
382 void disp_menu(MENU *mb)
383 {
384      int i,j;
385
386      for(i=1;i<= mb->num_row;i++)
387         {
388           for(j=1;j<=(mb->num_col);j++)
389              {
390                if((i-1)*mb->num_col+j<=mb->total)
391                   dispxynstr(mb->top_x+(j-1)*(mb->width+mb->space),
392                              mb->top_y+i-1,mb->width,
393                              mb->item[mb->num_col*(i-1)+j-1]);
394                else
395                   return;
396              }
397         }
398      for(i=1;i<=mb->total;i++)
399        disp_bar(mb->top_x+((i-1) % mb->num_col)*(mb->width+mb->space),
400                 mb->top_y+((i-1) / mb->num_col),(i!=mb->total) ?
401                 mb->width+mb->space : mb->width,mb->attr_c);
402 }
403
404 /*-------------------------------------------------------------------*/
405
406 void disp_bar(int x,int y,int width,int attr)
407 {
408      pokexynstra(x,y,width,attr);
409 }
410
411 /*-------------------------------------------------------------------
412
413 menubar         Menu bar
```

▶

```
414
415 Usage          void menubar (int *key,int *choice,MENU * mb);
416
417 Description    Menu bar function display choice which be selected
418               and read key to move bar . The key and choice will
419               modify when it return .
420
421 Return value   no return valu
422
423 */
424
425 void menubar(int *key,int *choice,MENU * mb)
426 {
427     int selecte_key;
428     int cur_choice,tchoice;
429
430     cur_choice=*choice;
431     disp_menu(mb);
432     disp_bar(mb->top_x+((*choice-1) % mb->num_col)*(mb->width+mb->space),
433            mb->top_y+((*choice-1) / mb->num_col),mb->width,mb->attr_b);
434     while(1)
435       {
436         selecte_key=getkey();
437         if(selecte_key==CR || selecte_key==ESC)
438            {
439              *key=selecte_key;
440              return;
441            }
442          if((selecte_key >= HOMEKEY && selecte_key <= PGDNKEY) ||
443              selecte_key == CTRL_E    || selecte_key == CTRL_X ||
444              selecte_key == CTRL_A    || selecte_key == CTRL_F ||
445              selecte_key == CTRL_C    || selecte_key == CTRL_R ||
446              selecte_key == BLANK_KEY || selecte_key ==TAB_KEY ||
447              selecte_key == SHIFT_TAB)
448             {
449              switch(selecte_key)
450                  {
451                    case HOMEKEY: *choice=1;
452                                  break;
453
454                    case ENDKEY : *choice=(mb->num_row-1)*(mb->num_col)+1;
455                                  break;
456
457                    case BLANK_KEY:
458                    case CTRL_X :
459                    case DOWNKEY: (*choice)+=mb->num_col;
460                                  if(*choice > mb->total)
461                                     *choice=1;
462                                  break;
463
464                    case CTRL_E :
465                    case UPKEY  : (*choice)-=mb->num_col;
466                                  if(*choice <=0)
467                                     *choice=mb->total;
468                                  break;
469
470                    case CTRL_A :
```



```
471                    case SHIFT_TAB:
472                    case LEFTKEY: (*choice)-=((mb->num_col==1) ? 0 : 1);
473                                  if(*choice <= 0)
474                                     *choice=mb->total;
475                                  break;
476
477                    case CTRL_F :
478                    case TAB_KEY:
479                    case RIGHTKEY:(*choice)+=((mb->num_col==1) ? 0 : 1);
480                                  if(*choice > mb->total)
481                                     *choice=1;
482                                  break;
483
484                    case PGDNKEY: *choice=mb->total;
485                                  break;
486
487                    case PGUPKEY: *choice=mb->num_col;
488                  }
489               disp_bar(mb->top_x+((cur_choice-1) % mb->num_col)*(mb->width+mb->spa
   ce),
490                        mb->top_y+((cur_choice-1) / mb->num_col),mb->width,mb->attr
   _c);
491               disp_bar(mb->top_x+((*choice-1) % mb->num_col)*(mb->width+mb->space)
   ,
492                        mb->top_y+((*choice-1) / mb->num_col),mb->width,mb->attr_b)
   ;
493               cur_choice=*choice;
494             }
495           else
496             {
497               if(strlen(mb->hkey)!=0)
498                  {
499                    selecte_key=_toupper(selecte_key);
500                    tchoice=0;
501                    if((tchoice=(unsigned int)strchr(mb->hkey,selecte_key))!=0)
502                        {
503                          tchoice-=(unsigned int)mb->hkey;
504                          *key=selecte_key;
505                          *choice=tchoice+1;
506                          disp_bar(mb->top_x+((cur_choice-1) % mb->num_col)*(mb->wi
   dth+mb->space),
507                                   mb->top_y+((cur_choice-1) / mb->num_col),mb->wid
   th,mb->attr_c);
508                          disp_bar(mb->top_x+((*choice-1) % mb->num_col)*(mb->width
   +mb->space),
509                                   mb->top_y+((*choice-1) / mb->num_col),mb->width,
   mb->attr_b);
510                          return;
511                        }
512                    }
513             }
514
515       }
516 }
517
518 /*-------------------------------------------------------------*/
519
```